Part 7

The Transformations

The Model

- The AST/Model is a critical part

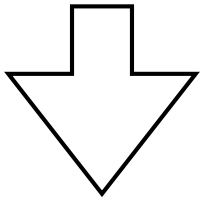
```
print 3 + 4 * 5 + x;

PrintStatement(
    Bin('+',
        BinOp('+',
              Integer(3),
              BinOp('*', Integer(4), Integer(5))),
        Location('x')))
```

- Question: Can the model be simplified?

Model Transforms

- Sometimes you can translate...

```
PrintStatement(
    Bin('+',
        BinOp('+',
              Integer(3),
              BinOp('*', Integer(4), Integer(5))),
        Location('x')))
```



```
PrintStatement(
    Bin('+', Integer(23), Location('x')))
```

- Example: Constant folding

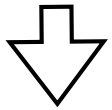
Insight

- Sometimes complicated language features can be expressed in terms of simpler language features (as a kind of rewriting)
- Related: "Syntactic Sugar"
- Might also be able to perform optimizations

Example

- Common subexpression elimination:

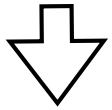
```
(x+y)/2 + (x+y)/4
```



```
{ var t = x + y;  t/2 + t/4; }
```

- Short-circuit evaluation

```
x && y
```



```
{ var t = x;  if t { t = y;} t; }
```

Implementation

- Pattern...

```
def transform(node):
    ...
    return new_node

def transform_binop(node):
    left = transform(node.left)
    right = transform(node.right)
    # Decide what to do ...
    if (isinstance(left, Integer) and
        isinstance(right, Integer)):
        return Integer(
           eval(f'{left.value} {node.op} {right.value}')
        )
    return BinOp(node.op, left, right)
```

Project

- You can optionally try to implement some transforms for Wabbit
- Example: constant folding
- Example: short-circuit
- See wabbit/transform.py